IN THE CLAIMS

Please amend the claims as indicated below:

1. (canceled)

2. (currently amended) A method of optimizing ~~the~~ compiled code generated from a high level computer programming ~~languages~~, wherein the compiled code includes loop constructs, the method comprising the steps:

(a) storing a predetermined number n for a compiler;

(b~~1~~) providing a non-optimized loop code segment corresponding to a loop construct written in a high level programming language, wherein in the non-optimized loop code segment the loop construct may be ~~is~~ executed at run time at least a loop repetition number of times n, depending upon a value at run time of a variable in a loop termination condition of the loop code segment, and the non-optimized loop code segment includes a call to a procedure;

(c~~2~~) providing a non-optimized pre-loop code segment corresponding to programming instructions preceding the loop construct, and a non-optimized post-loop code segment corresponding to instructions succeeding the loop construct;

(d~~3~~) ~~providing execution conditions such that program variables of the compiled code have certain values satisfying initial and terminating conditions for causing execution of the loop construct the loop repetition number of times n,~~ performing a computation by the compiler for determining $n+1$ values for the variable in the loop termination condition, such that a first one of the values would, upon execution at run time, terminate the loop code segment in 0 iterations, a second one of the values would, upon execution at run time, terminate the loop code segment in 1 iteration, and an $(n+1)$th one of the values would, upon execution at run time, terminate the loop code segment in n iterations, wherein the call invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables;

~~(4) revising the non-optimized pre-loop, loop and post-loop code segments to include the execution conditions; and~~

(e~~5~~) optimizing the non-optimized pre-loop, loop and post-loop code segments, wherein the optimizing includes the compiler:

propagating the values to the loop code segment, the pre-loop code and the post-loop code;

detecting effects on the loop code segment, the pre-loop code and the post-loop code of propagating the values, including detecting redundancies and invariant expressions in the loop code segment, the pre-loop code and the post-loop code resulting from the propagation of the values, so that the compiler may eliminate, or at least simplify, some of the loop code segment, pre-loop code and post-loop code for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*, wherein the consolidated code includes certain code of the non-optimized loop code segment, the non-optimized pre-loop code and the non-optimized post-loop code and omits certain other code of the non-optimized loop code segment, the non-optimized pre-loop code and the non-optimized post-loop code and wherein the call is omitted from the consolidated loop code segment if the certain program variable values are such that the call invoking the procedure results in no change in program variables;

(f6) determining whether the consolidated code segment should be executed in preference to the non-optimized code segments; and

(g7) if said determination is favourable, including the consolidated code segment in optimized code for a program written in the high level programming language.

3 - 6. (canceled)

7. (currently amended) The method as claimed in claim 2, wherein said step (f) determination involves a cost-benefit analysis to determined whether the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

8. (original) The method as claimed in claim 2, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

9. (currently amended) The method as claimed in claim 2, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

10 - 11. (canceled)

12. (currently amended) A compiler for optimizing the compiled code generated from high level computer programming languages wherein the compiled code includes loop constructs, the compiler being embodied on a computer-readable medium, the compiler comprising:

    <u>compiler code means for storing a predetermined number n for a compiler;</u>

    ~~1)~~compiler code means for providing a non-optimized loop code segment corresponding to a loop construct written in a high level programming language, wherein in the non-optimized loop code segment the loop construct <u>may be</u> ~~is~~executed <u>at run time at least</u> a loop repetition number of times n <u>, depending upon a value at run time of a variable in a loop termination condition of the loop code segment,</u> and the non-optimized loop code segment includes a call to a procedure;

    ~~2)~~compiler code means for providing a non-optimized pre-loop code segment corresponding with programming instructions preceding the loop construct, and a non-optimized post-loop code segment corresponding with instructions succeeding the loop construct;

    ~~3)~~compiler code means for ~~providing execution conditions such that program variables of the compiled code have certain values satisfying initial and terminating conditions for causing execution of the loop construct the loop repetition number of times n,~~ <u>performing a computation by the compiler for determining $n+1$ values for the variable in the loop termination condition, such that a first one of the values would, upon execution at run time, terminate the loop code segment in 0 iterations, a second one of the values would, upon execution at run time, terminate the loop code segment in 1 iteration, and an $(n+1)$th one of the values would, upon execution at run time, terminate the loop code segment in n iterations,</u> wherein the call invokes the procedure responsive to arguments of the call, the arguments including ones of the program variables;

    ~~4) compiler code means for revising the non-optimized pre-loop, loop and post-loop code segments to include the execution conditions; and~~

<div align="center">Page 4 of 10</div>

5) compiler code means for optimizing the non-optimized pre-loop, loop and post-loop code segments, wherein the optimizing includes the compiler:

propagating the values to the loop code segment, the pre-loop code and the post-loop code:

detecting effects on the loop code segment, the pre-loop code and the post-loop code of propagating the values, including detecting redundancies and invariant expressions in the loop code segment, the pre-loop code and the post-loop code resulting from the propagation of the values, so that the compiler may eliminate, or at least simplify, some of the loop code segment, pre-loop code and post-loop code for the execution conditions to provide a consolidated code segment corresponding with the execution conditions for execution of the loop said loop repetition number of times *n*, wherein the consolidated code includes certain code of the non-optimized loop code segment, the non-optimized pre-loop code and the non-optimized post-loop code and omits certain other code of the non-optimized loop code segment, the non-optimized pre-loop code and the non-optimized post-loop code and wherein the call is omitted from the consolidated loop code segment if the certain program variable values are such that the call invoking the procedure results in no change in program variables;

6) compiler code means for determining whether the consolidated code segment should be executed in preference to the non-optimized code segments; and

7) compiler code means for including the consolidated code segment in optimized code for a program written in the high level programming language, if said determination is favourable.

13 - 16. (canceled)

17. (currently amended) The compiler as claimed in claim 12, wherein said determination determining whether the consolidated code segment should be executed in preference to the non-optimized code segments includes determining by involves a cost-benefit analysis to determined whether the cost of using the consolidated code segment is reduced by a predetermined threshold compared with not using the consolidated code segment.

Page 5 of 10

18. (original) The compiler as claimed in claim 12, wherein the inclusion of said consolidated code segment in the optimized code is conditional on the occurrence of the execution conditions.

19. (currently amended) The compiler as claimed in claim 12, wherein said loop constructs includes any one or more of the following loop constructs: for loops, while loops, repeat loops.

20. (canceled)